College of Engineering and Computer Science
Mechanical Engineering Department

California State University Northridge

Mechanical Engineering 309
Numerical Analysis of Engineering Systems

Spring 2014 Number: 15237 Instructor: Larry Caretto

Solutions to Programming Assignment Five – Interpolation and Numerical Differentiation

Unless stated otherwise use the "standard" data set below for all interpolation problems in this assignment.

x	805	825	845	865	885	905	925	945	965	985
y	0.710	0.763	0.907	1.336	2.169	1.598	0.916	0.672	0.615	0.606

1. Using MATLAB

a. Review the help file for the MATLAB spline function. Copy the standard data set above into MATLAB and plot a curve showing a comparison of the data and a cubic spline fit. (Obtain plot data for $\Delta x = 1$.) Compare this plot to the one in the course notes. Discuss this comparison and list the option that you used in MATLAB for handling the endpoints.

According to the MATLAB documentation using the spline function in the form spline(x,y,xx) where x and y are equal sized vectors results in the "not-a-knot" end condition. This condition was derived in class in two ways (1) by assuming that the second derivative is linear near the endpoint and (2) by assuming continuity of the third derivative at the second and next-to-last knots. Other end conditions are possible as described in the help section. MATLAB also has an optional" Curve-Fitting" toolbox, that contains a splinetool function, which allows different end conditions. For a discussion of spline end conditions, see the following websites for more information about MATLAB curve fitting: http://www.mathworks.com/matlabcentral/newsreader/view_thread/172988. See http://www.mathworks.com/help/toolbox/curvefit/splinetool.html for information about splinetool.

b. Review the help file for the MATLAB function polyfit. Obtain a 9th order polynomial to fit the same data used for the spline fit. Use the transformed x variable to reduce error as described in the help file for polyfit. Prepare a plot that compares the interpolated results with the original data. . (Obtain plot data for $\Delta x = 1$.) Discuss any difference between your results and those in the course notes.

The MATLAB code for both parts a and b is shown below.

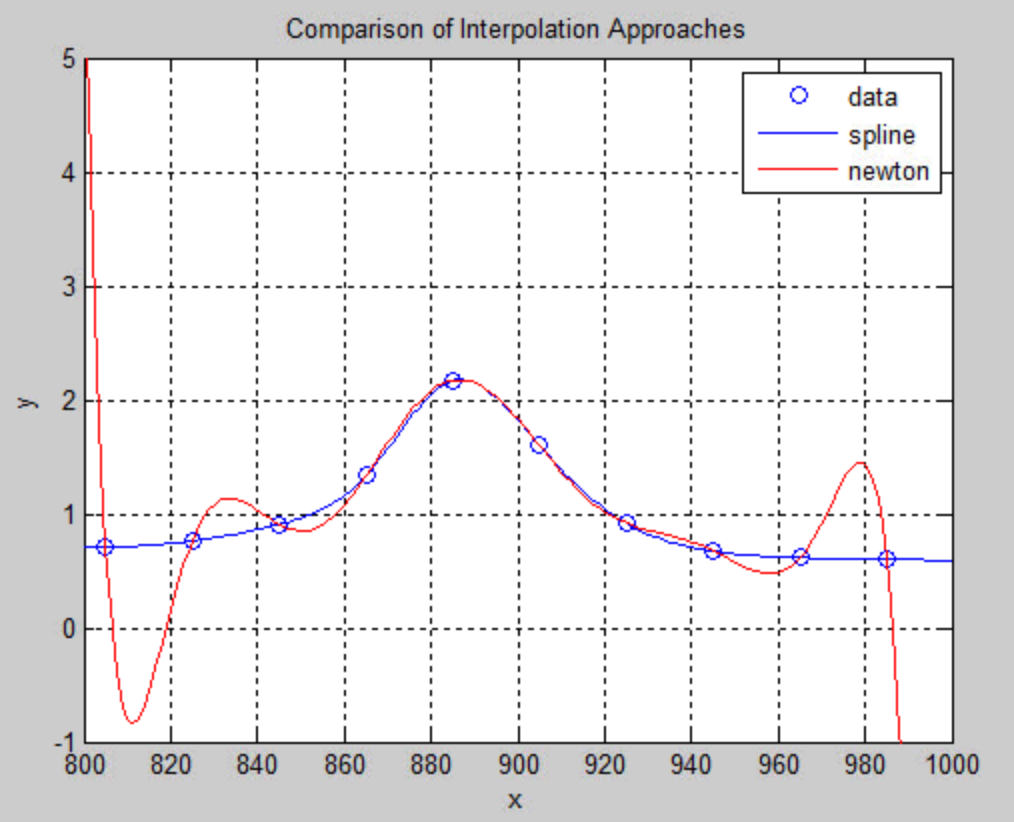
```
>> %Initial code that applies to both spline and polynomial fits
>> format compact
>> %Get x and y data for interpolation
>> x = [805 825 845 865 885 905 925 945 965 985];
>> y = [0.710 0.763 0.907 1.336 2.169 1.598 0.916 0.672 0.615 0.606];
>> %Get an xPlot array to plot interpolated functions
>> xPlot = 800:1000;
>> %Get ySpline to plot as y values fitted to spline interpolation
>> ySpline = spline(x,y,xPlot);
>> %Get scaled coefficients for Newton polynomial
>> [p S mu]=polyfit(x,y,9);
>> %Get y data to plot using polyval arguments to scale xPlot
>> yNewton = polyval(p,xPlot,S,mu);
>> plot(x,y,'o',xPlot,ySpline,'b',xPlot,yNewton,'r')
>> axis([800 1000 -1 5])
>> legend('data', 'spline', 'newton')
```

Jacaranda Hall 3314 Mail Code Phone: N/A
Email: lcaretto@csun.edu 8348 Fax: 818.677.7062

```
>> grid     %places grid on plot
>> title('Comparison of Interpolation Approaches')
>> xlabel('x')
>> ylabel('y')
```

The resulting MATLAB plot is shown in Figure 1, below. The plot appears to be almost the same as the cubic spline and Newton polynomial plots shown in the class notes. The spline plot gives a smooth fit to the data while the 9th-order Newton polynomial has unrealistic overshoots and undershoots of the data range. Near the maximum and minimum values of the x data used for the interpolation the cubic spline appears to give reasonable extrapolation results, while the Newton polynomial increases dramatically below the first x data point and decreases just as dramatically as the value of x goes beyond the final data point used for the interpolation.

Figure 1 – Copy of MATLAB plot



2. Using EXCEL VBA

The results from parts 2 and 3 are summarized below. Download the workbook pa5soln.xlsm to see all the cell formulas and VBA code for both these parts.

a. Write a VBA function that allows users to select x and y data from a table in an Excel worksheet that they want to use as an interpolation table and, in the same function select another x value (not in the interpolation table) for which they want an interpolated value. You can assume that the input x and y data for the interpolation will be data ranges of 2 or more cells each; you can also assume that these cells will be in rows or in columns. The number of cells the user selects will determine the order of the polynomial. Your function should ensure that the user has selected the same number of cells for both the x and the y data.

Apply your VBA code to obtain interpolation results for x = 853 and 962 using linear, quadratic and cubic polynomials that properly select the correct input data to place these

x values near the center of the range for the interpolation data. Discuss the differences in your results for each interpolated x value.

The results are shown in the table below. This table is taken from the workbook, pa5soln.xlsm; the VBA code is also in this workbook.

The cell formula used to determine these results has the following form: newton(<xRange>, <yRange>, <xPoint>). In this formula <xRange> is a set of n column cells giving the x data for the interpolation formula; <yRange> is the set of n column cells for the y data that corresponds to the x data selected previously; <xPoint> is a single value of x to which the interpolation formula (determined by the data in <xRange> and <yRange>) will be applied. The order of the interpolation formula is determined by number of cells selected by the user. Selecting n cells gives an interpolation polynomial of order n – 1.

Users should select data that places the value of <xPoint> in the middle of the range used for the interpolation data. If the uses wants a polynomial of an odd order (1, 3, 5, …) she will have to select an even number of data points. These should be split so that <xRange> (and the corresponding <yRange> contain the same number of points on each side of <xPoint>. If the polynomial has an even order, the selection is more difficult because there will be more data points on one side of <xPoint> than on the other. In this case the user should first consider an even number of points ranging from <xRange(1)>| to <xRange(n)>, with the name number of points on each side of <xPoint>. Ths user should then discard the data point that has the higher value of |<xPoint> – <xRange(1)>| or |<xPoint> – <xRange(n)>|. This criterion was used to select the data points for the results in the "Quadratic" column in the table below. In that column, the data points at x = 825, 845, and 865 were used to interpolate the value at x = 853 and the values at x = 945, 965, and 985 were used to interpolate the value at x = 962. In the "Quadratic2", the choices for the first interpolation were the points at x = 845, 865, and 885, and, for the second interpolation, the values at x = 925, 945, and 965. These choices are seen to have only a small effect on the interpolated values.

Interpolation poly-nomial Results		Polynomials Used			
		Linear	Quadratic	Quadratic2	Cubic
Interpolation x Values	853	1.079	1.044	1.030	1.036
	962	0.624	0.620	0.612	0.617

b. Use the LINEST function to find the regression coefficients for the data shown on the "Regression" tab of the workbook for this assignment and use them as described in the steps below.

i. Obtain the regression coefficients for y as a function of three variables, x_1, x_2, and x_3; identify the regression coefficients and their standard errors on the worksheet.

These results, copied from the worksheet, are shown below.

LINEST Function Results				
	Slopes			Intercept, b_0
	b_3 for x_3	**b_2 for x_2**	**b_1 for x_1**	
	0.000654	**9.03E-05**	**0.2320**	1.1449
Standard Errors for first row	0.000826	0.000771	0.1170	0.4197
R^2 and $s_{y\|x}$ values	0.04055	0.8353	#N/A	#N/A
F statistic/degrees of freedom	2.3245	165	#N/A	#N/A
Sums of squares	4.8659	115.13	#N/A	#N/A

ii. Once you have obtained these coefficients, use them to compute the estimated y values for all the data points. Use these estimated values to compute the R^2 coefficient for the regression and compare it to the value given by LINEST.

These estimated values of y (yhat) are computed as $1.1449 + 0.2320x_1 + 9.03x10^{-5}x_2 + 0.000654x_3$. The mean value of y is found to be 2.18, using the average function of Excel. Once these values are found the R^2 term is computed from the usual equation, shown below.

$$R^2 = 1 - \frac{\sum_{m=1}^{N}(y_m - \hat{y}_m)^2}{\sum_{m=1}^{N}(y_m - \bar{y})^2}$$

The computed value agrees with the one produced by LINEST. The value of $R^2 = 0.04055$ shows that this is not a good fit!

iii. Use the TINV function with a confidence level of 95% to find the confidence limits for the regression coefficients.

The confidence intervals for the regression coefficients are for a two-sided test. The Excel calculation of the t-statistic for a two-sided test, at any confidence level, L, is found by the function call T.INV.2T(1-L,df) [or the compatibility function TINV(1-L,df)], where df is the degrees fo freedom. For a 95% confidence level and the degrees of freedom returned by linest we use the function call TINV(0.05, 165); this returns a value of 1.974. This is multiplied by the standard error, giving the following ranges for the slopes: $b_0 = 1.145 \pm 2.261$, $b_1 = 0.2320 \pm 0.4581$, $b_2 = 9.03x10^{-5} \pm 17.8x10^{-5}$, $b_3 = 0.000654 \pm 0.00129$. The fact that these confidence levels are larger than the values of the regression coefficients is another indicator that this is a poor fit.

3. The following assignment can be done on an Excel worksheet or in the MATLAB command window, as you prefer.

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2h} + O(h^2) \qquad f_i' = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12h} + O(h^4)$$

$$f_i'' = \frac{f_{i-1} - 2f_i + f_{i+1}}{h^2} + O(h^2) \qquad f_i'' = \frac{-f_{i-2} + 16f_{i-1} - 30f_i + 16f_{i+1} - f_{i+2}}{12h^2} + O(h^4)$$

Use each formula to compute the derivatives of cos(x) at x =1 for a range of h values from 0.1 to 10^{-10}. Plot the errors (the difference between the numerical result and the correct value of dcos(x)/dx = -sin(π/2)) on a log-log plot and verify that the order of the error is correct. Identify the locations on the plot where the roundoff error begins to occur.

The table and graph below show the typical behavior of the error for the given case at x = 1. The shaded areas in the table show the areas where roundoff error plays a role and the relationship between error and order is not followed.[1] The relationship between order and error is shown by the table entries for larger step sizes (below the shaded areas.)

The table also shows unusual results for x = π/2. This calculation was not required for the assignment; it is shown here because of the unusual results for the second derivative. For the cosine, $d^2\cos(x)/dx = -\cos(x)$. This is zero at x = π/2. The numerical computation of this derivative uses the zero value of cosine at x = π/2 and the remainder of the computation uses values at π/2 ± kh (where k = 1 or 2). The values at π/2 ± kh for any k are opposite in sign and their addition in the finite difference expression causes them to cancel giving the correct value of zero for almost any step size. In this case, the expected relationship between step size and error does not occur. This example is included in this solution as a warning that finite-difference results may have exceptional cases where the relationship between order and step size does not occur.

[1] Download the workbook pa5soln.xlsm to see the actual worksheet calculations.

The results for the various cases can be summarized as follows:

- For the second-order first derivative at x =1 and x = π/2, between h = 0.1 and h = 0.001, the error decreases by a factor of 10^2 as the step size is cut by a factor of 10. (At x = π/2, this behavior continue to a step size of 0.0001.)
- For the fourth-order first derivative, at both x = 1 and x = π/2, between h = 0.1 and h = 0.01, the error decreases by a factor of 10^4 as the step size is cut by a factor of 10.
- For the second-order second derivative at x = 1, between h = 0.1 and h = 0.001, the error decreases by a factor of 10^2 as the step size is cut by a factor of 10.
- For the fourth-order second derivative at x = 1, between h = 0.1 and h = 0.01, the error decreases by a factor of 10^4 as the step size is cut by a factor of 10.

Error Results for Cosine Derivatives

Step size	Errors Values at x = π/2				Errors Values at x = 1			
	First Derivatives		Second Derivatives		First Derivatives		Second Derivatives	
	2nd Order	4th Order	2nd Order	4th Order	2nd Order	4th Order	2nd Order	4th Order
1E-10	8.27E-08	8.27E-08	6.13E-17	6.13E-17	2.03E-07	3.88E-07	5.40E-01	5.40E-01
1E-09	8.27E-08	1.20E-07	6.13E-17	6.13E-17	1.92E-08	1.78E-08	2.22E+02	2.96E+02
1E-08	6.08E-09	9.78E-09	6.13E-17	6.13E-17	3.03E-09	3.03E-09	5.40E-01	5.40E-01
1E-07	5.84E-10	9.54E-10	6.13E-17	6.13E-17	3.06E-10	6.76E-10	1.48E-02	2.22E-02
1E-06	8.24E-11	1.19E-10	6.13E-17	6.13E-17	2.75E-11	6.45E-11	6.78E-05	2.90E-04
1E-05	1.01E-11	1.03E-11	6.13E-17	6.13E-17	1.09E-11	2.09E-12	3.27E-06	3.27E-06
1E-04	1.67E-09	1.10E-13	6.13E-17	6.13E-17	1.40E-09	1.32E-13	3.03E-08	4.51E-08
1E-03	1.67E-07	1.80E-13	6.13E-17	1.80E-14	1.40E-07	5.78E-14	4.52E-08	1.57E-10
1E-02	1.67E-05	3.33E-10	6.57E-15	6.75E-15	1.40E-05	2.80E-10	4.50E-06	5.96E-11
1E-01	1.67E-03	3.33E-06	3.00E-16	2.44E-16	1.40E-03	2.80E-06	4.50E-04	6.00E-07